Syntax Editing for Mark IV-A System Performance Test Software

G. N. Jacobson
Operations Sustaining Engineering Section

This article describes the syntax editing concepts used by the Operations Sustaining Engineering Section in implementing System Performance Test software for the Mark IV-A era. The processing functions are discussed, as well as the necessary data structures and table generation macros used in implementing those functions. In addition, the procedural and software interfaces which have been developed for users of the syntax editor are described, including the forms required for establishing directive and parameter characteristics.

I. Introduction

The System Performance Test (SPT) software package being developed by the Operations Sustaining Engineering Section for the Mark IV-A DSN implementation will reside in the backup Complex Monitor and Control (CMC) computer, a Modcomp 7845. It consists of a test executive and a set of six application tasks. Basically, the executive distributes input data to the applications, provides resource allocation services, and performs common processing such as data block dumping, display generation, test procedure reading, and syntax editing. The application tasks are each designed to test a particular DSN system. Tracking, Telemetry, Command, Monitor and Control, Very Long Baseline Interferometry, and Radio Science will all be supported by SPT software for the Mark IV-A configuration. (Frequency and Timing will not require SPT software for performance testing.)

Approximately 300 SPT input directives will be required to control the SPT software. These directives may be entered by the operator or may be read from a test procedure file residing on disk. In prior versions of SPT software, each application developed its own syntax editing functions, based on the formats of its own directives. For Mark IV-A, the bulk of the syntax checking will be done by the SPT test executive.

To accomplish this, there is a set of resident subroutines that perform common syntax editing functions such as limit checking, default assignments, existence of required fields, illegal characters, etc. In addition, there is a set of subroutines that perform functions peculiar to a single directive or a small group of directives. These subroutines are referred to as Individual Directive Processors (IDPs). If the directive has passed all the checks made by the syntax editor, then the appropriate IDP will be executed. The IDP can make additional syntax checks pertinent only to this directive (if applicable). If the directive is syntactically correct, the IDP will then perform the intended function of the directive or set the applicable flag(s) for use by the applications task.

II. Syntax Editor Output Buffer

The primary interface between the resident syntax editor and the IDPs is the 100-word syntax editor output buffer. The first six words of the buffer contain execution time, the overall status of the syntax check, the number of words in the buffer required for this particular directive, and the CAN code (compressed alpha-numeric code) for the directive name. The remainder of the buffer is defined by the programmer implementing the directive. The detailed contents of the remaining words in the buffer are itemized on the Directive Characteristics Description form (see Fig. 1.) The programmer defines a canonical form of the directive. This means that all fields are shown on the Directive Characteristics Description form and that they appear in the same sequence as desired for the output buffer. The output buffer contains a parameter status word for each parameter. The parameter itself is passed in a format specified by the user.

To specify the characteristics of each parameter in detail, a Field Characteristics Description form is used (see Fig. 2.) The entries made on this form determine the actual syntax checks which will be made by the syntax editor. A field listed as required must obviously be included with each occurrence of the directive. The field may be interrelated to other fields in the sense that at least one, exactly one, or at most one of a series of fields may be entered. This can be easily recorded on the Field Characteristics Description form and checked by the syntax editor.

The programmer must next define whether the field consists of a value, a self-identifying variable, or a variable identifier equalling a value. In the third case, there are two subfields involved, the variable identifier and the value. This is why the Field Characteristics Description allows two subfield entries per field.

The input format may be specified as alphanumeric, integer (including octal or hexadecimal), numeric (may include a decimal point), time (UTC or HMS format), text string, or expression. Man-machine interface standards provide for directives with the variable portion of the directive containing a text string. This is treated in SPT software as a single value with an input format type of "text string."

The range value column will be filled in with lower and upper boundaries if limit checks are to be performed by the syntax editor.

Legal output formats include CAN code (single or double word), floating point (double, triple, or quadruple word), fixed point (single, double, or quadruple precision), time (ASCII, deciseconds since beginning of year, or milliseconds since midnight), integer, and ASCII. In the latter case, a maximum length in bytes may also be specified.

The syntax editor is capable of checking a subfield against a list of acceptable entries. The valid entries are listed in the next-to-last column of the Field Characteristics Description. It is possible to request an indexed output format type. The syntax editor will output the index number corresponding to the position of the actual parameter in the list of acceptable entries.

If a subfield is to assume a default value when it is omitted from the directive, the desired default value is entered in the last column of the Field Characteristics Description.

III. Syntax Definition Tables

The information contained on the Directive Characteristics Description and the Field Characteristics Description must be transcribed into tables for subsequent use by the syntax editor. SPT software design requires three tables for this purpose.

A. Directive Definition Table

The Directive Definition Table contains a two-word entry per directive (see Fig. 3.) The first word of the two-word entry contains the CAN code for the three characters which identify the directive. (This is possible because all SPT directives consist of four characters, where the first character identifies the system under test, and the last three characters consist of a mnemonic name for the directive.) The second word contains a pointer to the appropriate entry in the Directive Characteristics Table.

B. Directive Characteristics Table

Each entry in the Directive Characteristics Table consists of two parts. The first part is fixed in length, consisting of two words per directive (see Fig. 4.) The first word contains a series of flags to indicate for which subsystems this directive is allowable, whether it contains an expression or a text field, and if it is a fixed position type of directive. The second word contains the overall output size in words and the maximum number of subfields.

The second part of each entry in the Directive Characteristics Table is variable in length, based on the number of parameters in the directive. Each parameter requires two words in this part of the table (see Fig. 5.) The first word contains a series of flags which relate to information specified on the Field Characteristics Description form such as whether the field is required or optional, whether a default value applies, whether a list of acceptable values has been supplied, whether the field is to be range-checked, and whether any relational characteris-

tics (such as mutually exclusive fields) apply. Also contained in this word is the index for this parameter into the syntax editor's output buffer. The second word points to the address of the appropriate entry in the Parameter Characteristics Table, where all the detailed information relative to this parameter is stored.

C. Parameter Characteristics Table

The Parameter Characteristics Table also contains a fixed length portion and a variable length portion per entry (see Fig. 6.) The first four words are required for each parameter characteristic entry. The first word contains a pair of codes to describe the input format and output format specified on the Field Characteristics Description form. The first eight bits of the second word contain the maximum length of an ASCII field (if specified) or the number of bits to the right of the binary point (if fixed point). The remainder of the second word contains the index into the Parameter Characteristics Table for the default value and the number of words required for the default value, if applicable. The third word contains the index of the lower limit and the number of words required for each of the two limit values, if range checking is desired. (The upper limit value will immediately follow the lower limit.) Similarly, the fourth word contains the index of the first word of the list of acceptable values and the number of words required for each element of the list. In this case, the total number of entries in the list will also be stored in the fourth word (in the first eight bits).

Following this comes the variable length portion of the Parameter Characteristics Table entry. If a default value has been specified for this parameter, it will be stored next in the table. If range checking is desired, the lower and upper limits will follow. Finally, if a list of acceptable values has been supplied, it will be stored at the end of each table entry. The elements in the list will be stored in the same order as specified by the user. This is because an indexed output type is available for such a parameter. As described earlier, this means that the syntax editor will return to the user an index corresponding to the position in the list of acceptable values where the actual parameter was located. The user is then able to use this index directly in subsequent processing. For this reason, the indexed output type is very commonly requested when a list of acceptable values is supplied for the input field.

IV. Syntax Table Generation Macros

There is actually a missing step in the transcription process from the syntax definition forms to the syntax definition tables. It must be possible to generate the very detailed bit-oriented tables efficiently and accurately. The route selected was the use of macro techniques (Ref. 1).

Nine macros have been created for the purpose of syntax editor table generation. Two macros are for initialization and termination of the table generation process. The remaining seven are used for creating specific entries or parts of entries for the three tables described in the preceding section.

Four of the macros are used for creating entries in the Parameter Characteristics Table. The first of these is in fact called the parameter characteristics macro and is used to begin the definition of the characteristics of a particular parameter. It must be followed by corresponding default, range, and acceptable value macros, as appropriate, if the parameter has an associated default, range check, or list of acceptable values. The macros must occur in this specific sequence for any parameter.

The directive name macro is used for generating a directive name in CAN code format and a pointer to the corresponding entry in the Directive Characteristics Table. The two words become a complete entry in the Directive Definition Table. This macro must occur after all corresponding parameter characteristics macros.

The final two macros are required for creating entries in the Directive Characteristics Table. The first of these, the directive characteristics macro, begins the generation of an entry. It is used for creating the fixed length portion of each Directive Characteristics Table entry. This macro must appear immediately after its related directive name macro and must be immediately followed by any related parameter pointer macros. The parameter pointer macro generates the two-word subentry per parameter in the Directive Characteristics Table. A parameter pointer macro is required for each possible subfield of the directive and must appear in the same sequence as in the directive's canonical form.

Figure 7 shows a portion of the actual macros used in the Mark IV-A SPT software system for generating the three required tables for the syntax editor.

V. Architectural Design

The syntax editor is used for checking both directives entered by the operator and directives read from a test procedure file. As such, it is actually invoked by two routines, the main directive processor control routine and a separate routine which processes directives from the procedure processor. The editor itself consists of 22 subroutines. Figure 8 contains a hierarchical tree for the editor.

The top level, of course, consists of the main control routine for the syntax editor. It in turn calls a series of subrou-

tines which performs initialization functions, validates the directive, retrieves and converts individual parameters, checks relational characteristics (such as mutually exclusive fields or existence of at least one or at most one of a series of fields), sets default values, and checks for required parameters.

Of these, the only major expansion is for the routine which retrieves and converts individual parameters, EOPARM. EOPARM itself consists primarily of a loop for processing each parameter in the directive. It calls a routine to obtain the next parameter (which is further modularized to retrieve alphanumeric, text, or numeric subfields); checks that the retrieved parameter is legal for the prior trailing character; and then calls EOFSUB to correlate the input parameter with the subfield data contained in the Directive Characteristics Table and Parameter Characteristics Table. EOFSUB contains several subroutines at three additional hierarchic levels to perform all of the necessary conversions, range checking, and acceptable value list searching.

If the directive passes all of the syntax editor's checks, the appropriate Individual Directive Processor is established and executed. The IDP often has additional syntax checks to make, peculiar to this particular directive. For example, certain directives or certain fields may be allowed only under one or more specific operating modes. Or some fields may become required if some other specific field has been entered. Depending on the directive entered, the IDP often has other functions to perform, such as setting flags or data in a global or private shared common area. To assist the programming staff in attending to all details in the design of an IDP, a checklist has been prepared of possible IDP processing functions. This is illustrated in Fig. 9.

VI. Implementation Sequence

The Mark IV-A SPT software system is being implemented in a continually cycling, demonstrable fashion (Ref. 2). A Network of Demonstrable Functions (NDF) has been prepared, defining approximately 170 individual steps. Each step will demonstrate one specific function or group of functions and will normally require implementation of around a dozen new subroutines.

Development of the syntax editing routines and IDPs adheres to the SPT implementation philosophy. Each routine will be developed when it is required for demonstration of one of the specific functions laid out in the NDF. As an example, the syntax editor routine for retrieving a text subfield will not be implemented until the first scheduled step which requires a directive involving a text subfield. The IDP for enabling a long loop test will not be implemented until work begins on the long loop path of the NDF. Figure 10 shows a partial list of SPT directives and the NDF step in which they will be implemented.

When the design of each step is completed, a review is held internal to the Operations Sustaining Engineering Section. To assist in reviewing the design of the IDP, and to convey the proper syntax (and purpose) of each directive to the ultimate users of SPT software, a standard format has been established for use in the Software Operator's Manual (SOM). Figure 11 illustrates the required sections for one of the SPT directives used for running a Command test sequence. It should be noted that the directive response messages listed at the end of each SOM directive description are unique to that directive or a small subset of directives. Common messages, such as standard acknowledgements or errors detected by the syntax editor (as opposed to the IDP), will be listed and described once in a separate section of the SOM.

VII. Concluding Remarks

At this point in time, 20 of the syntax editor's 22 routines have been developed and demonstrated. (The two routines for fixed point conversions are not required at this time on our implementation plan.) The editor has proven to be extremely reliable. Close to 100 steps on the Mark IV-A SPT Network of Demonstrable Functions have been demonstrated, and each one has exercised the capabilities of the syntax editor to varying degrees. During this interval, only a few minor anomalies related to the syntax editor have been detected.

Initial design of the syntax editor has proven to be extremely sound. Only one major new capability had to be added since the primary concepts were developed. This was the capability to have the syntax editor check for relational characteristics, such as mutually exclusive fields. The design of the editor was flexible enough to allow this to be added with comparative ease.

As the application tasks require new directives for upcoming steps on the NDF, the directives are incorporated into the system easily and efficiently. Design and implementation of most IDPs are also fairly straightforward. What used to be a significant, time-consuming task for each application has been reduced to a routine, standardized activity for Mark IV-A SPT software development.

References

1. Spinak, A., "Syntax Table Generation Macros," Internal Memorandum No. SSG-FY82-257, Bendix Field Engineering Corporation, Pasadena, Calif., July 14, 1982.

2. Jacobson, G. N., and Spinak, A., "Top Down Implementation Plan for System Performance Test Software," *TDA Progress Report 42-70,* Jet Propulsion Laboratory, Pasadena, Calif., pp. 190-199, Aug. 15, 1982.

DIRECTIVE CHARACTERISTICS DESCRIPTION DATE:_____

REVISION:_____

1. NAME:_____ 2. STEP:_____

4. FUNCTION: 3. PROGRAMMER:_____

5. OVERALL GENERAL FORMAT:

6. LEGITIMATE SUBSYSTEM PREFIXES: $\overline{C}$ $\overline{K}$ $\overline{T}$ $\overline{R}$ $\overline{M}$ $\overline{V}$

7. MAXIMUM EXPECTED NUMBER OF PARAMETER FIELDS:_____

8. SYNTAX OUTPUT BUFFER:

1	TIME MSB	MLBTIM	ILBOUT HEADER
2	TIME LSB		
3	OVERALL STATUS	ILBSTA	
4	OVERALL OUTPUT SIZE	ILBSIZ	
5	DIRECTIVE CAN CODE UPPER	MLBCAN	
6	DIRECTIVE CAN CODE LOWER		
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

9. FIXED POSITION INPUT (Y OR N): ____

10. OTHER CHARACTERISTICS:

Fig. 1. Directive characteristics description

FIELD CHARACTERISTICS DESCRIPTION

1. DIRECTIVE NAME:_____ 2. PROGRAMMER _____

3. FIELD NUMBER:_____ 4. FIELD OPTIONAL (R OR O):_____
(R = REQUIRED, O = OPTIONAL)

5. FUNCTION:

6. DESCRIBE FIELD INTERRELATIONSHIPS (IF ANY)

NEXT RELATED FIELD NUMBER _____

AT MOST ONE _____ EXACTLY ONE _____ AT LEAST ONE _____

SUB-FIELD	V,VI, SIV	INPUT FORMAT	RANGE VALUES	OUTPUT FORMAT	MAX. OUTPUT LENGTH	LIST OF ACCEPTABLE ENTRIES (IN INDEX ORDER)	DEFAULT
SF1							
SF2							

7. OTHER CHARACTERISTICS:

Fig. 2. Field characteristics description

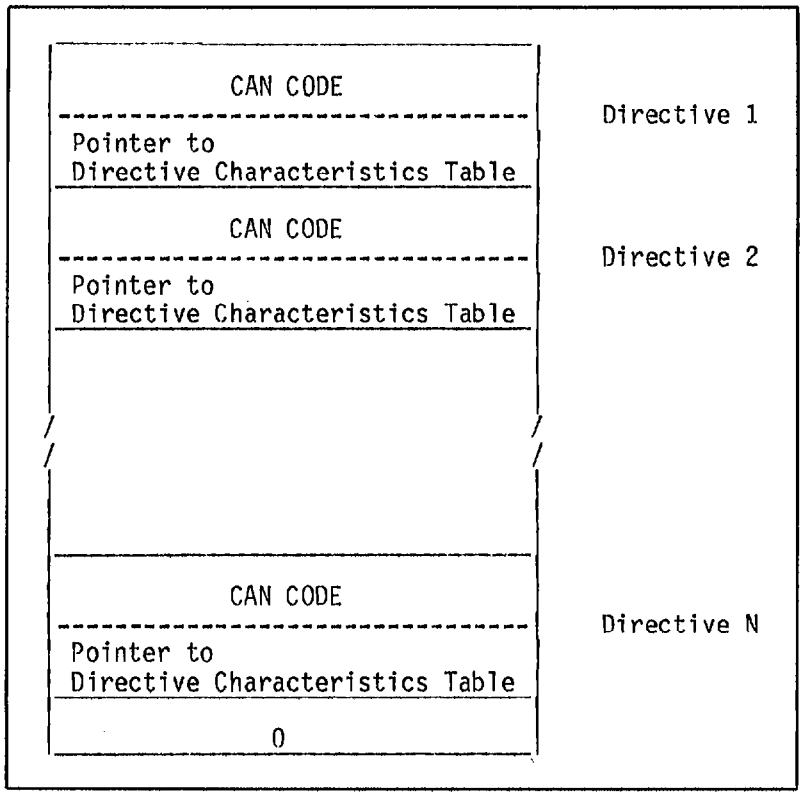

Fig. 3. Directive definition table

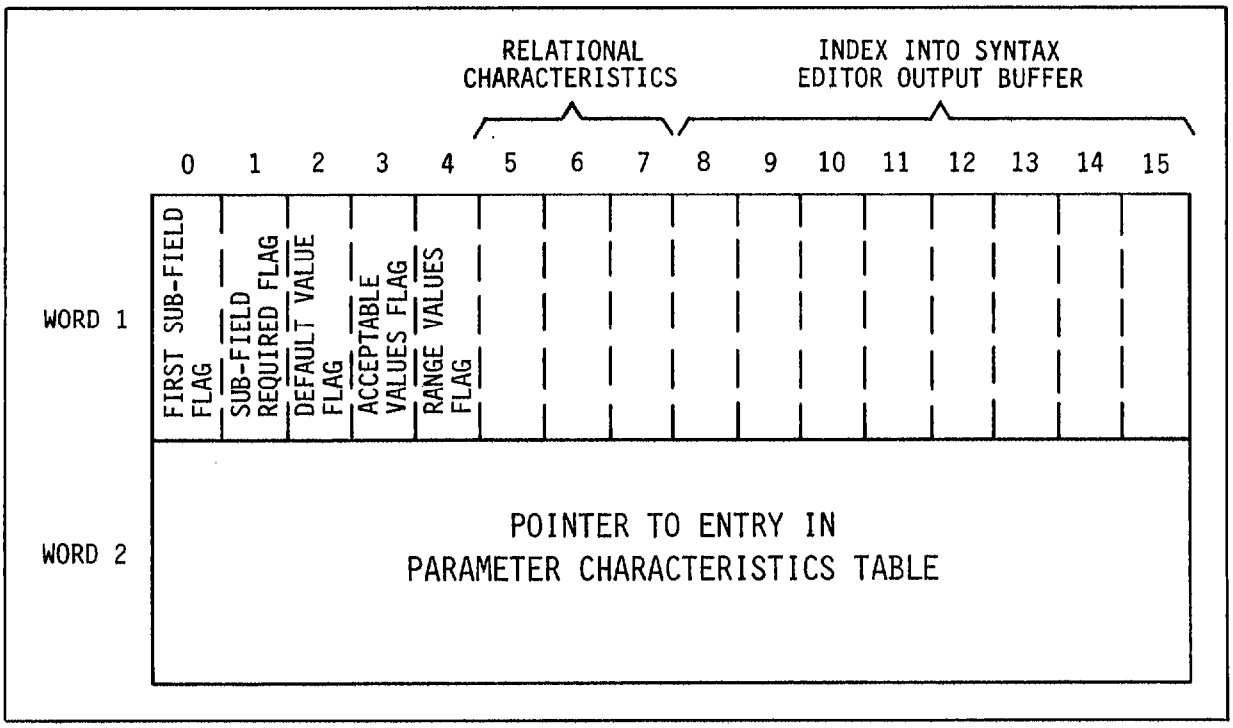

Fig. 4. Directive characteristics table, directive description entry

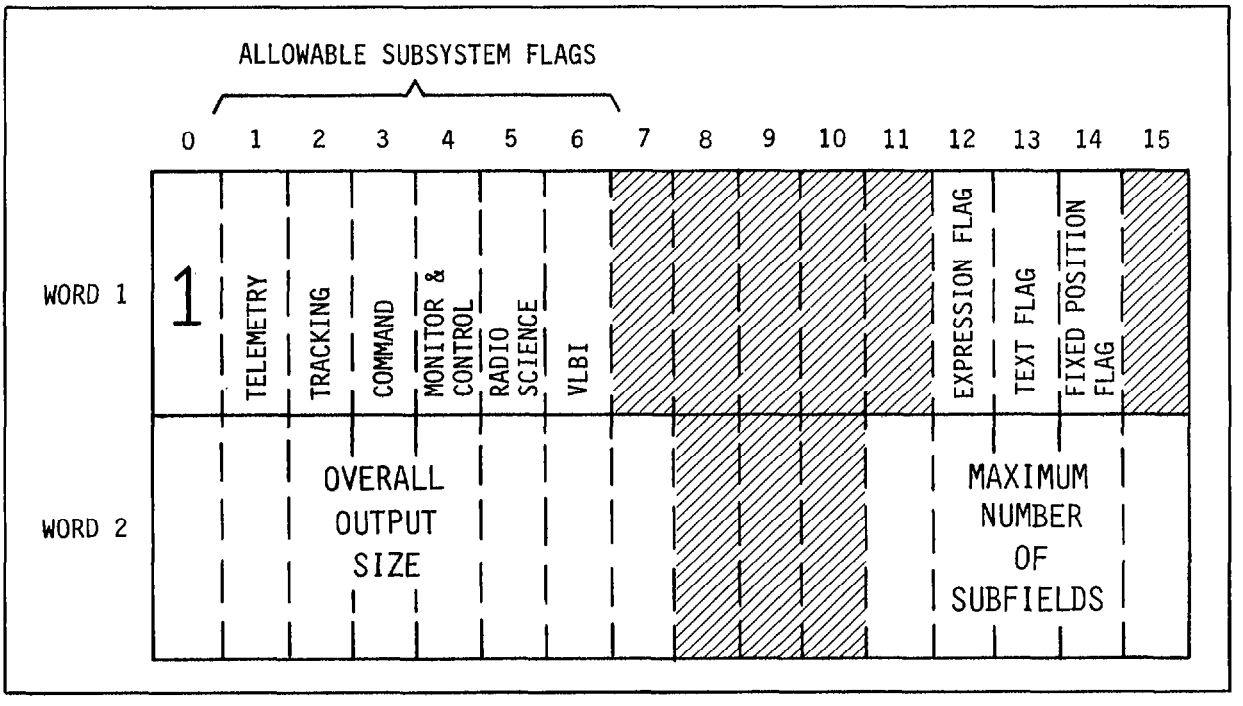

Fig. 5. Directive characteristics table, parameter description entry

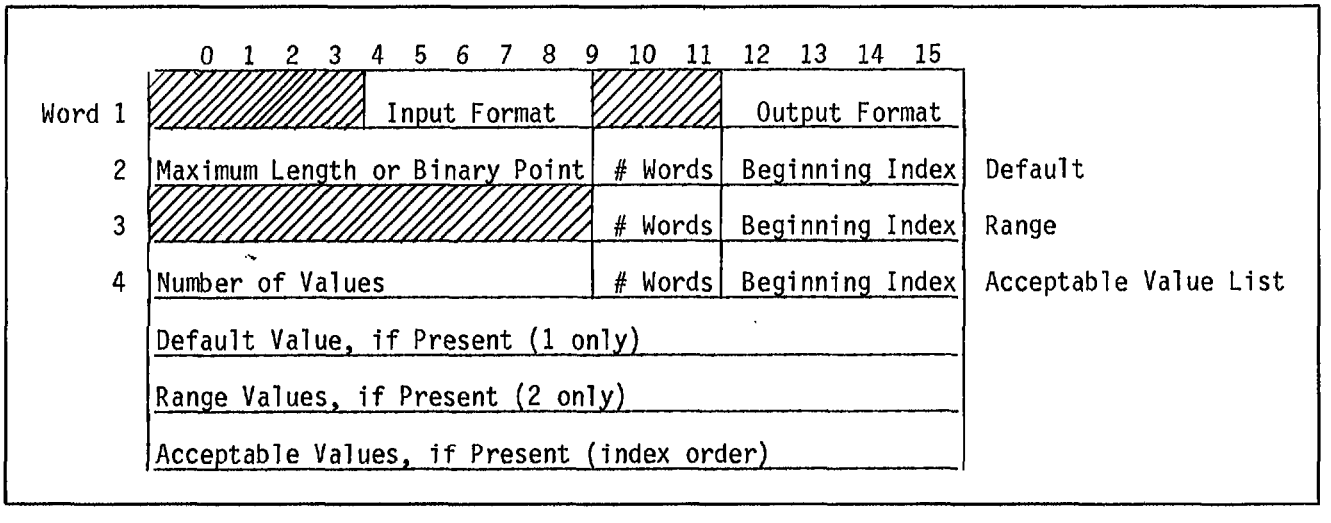

Fig. 6. Parameter characteristics table

```
*
*********************************************
* CAL Directive parameters                  *
*********************************************
PCAL1    PARCHR,2,,R      NUM,DF
         RANGES           1.0E1,1.0E3
*
PCAL2    PARCHR,2,A,,D    ALP,XX,2
         DEFAUL           1
         ACCVAL           "CFG","MON"
*
*********************************************
* CON Directive parameters                  *
*********************************************
PCON1    PARCHR,2,A       ALP,XX,4
         ACCVAL           "GET","SAVE","LIST","DEL"
*
PCON2    PARCHR,2,A       ALP,NP,1
         ACCVAL           "FILE"
*
PCON3    PARCHR           ALP,PT,,8
*


*
*********************************************
* CAL Directive                             *
*********************************************
         DIRNAM           CAL
         DIRCHR,T         11,2
         PARPTR,PCAL1,6   FS,R
         PARPTR,PCAL2,9   FS,D,A
*
*********************************************
* CON Directive                             *
*********************************************
         DIRNAM           CON
         DIRCHR,T         14,3
         PARPTR,PCON1,6   FS,SR,A
         PARPTR,PCON2,8   FS,SR,A
         PARPTR,PCON3,9   SR
*
```

Fig. 7. Sample SPT syntax table generation macros

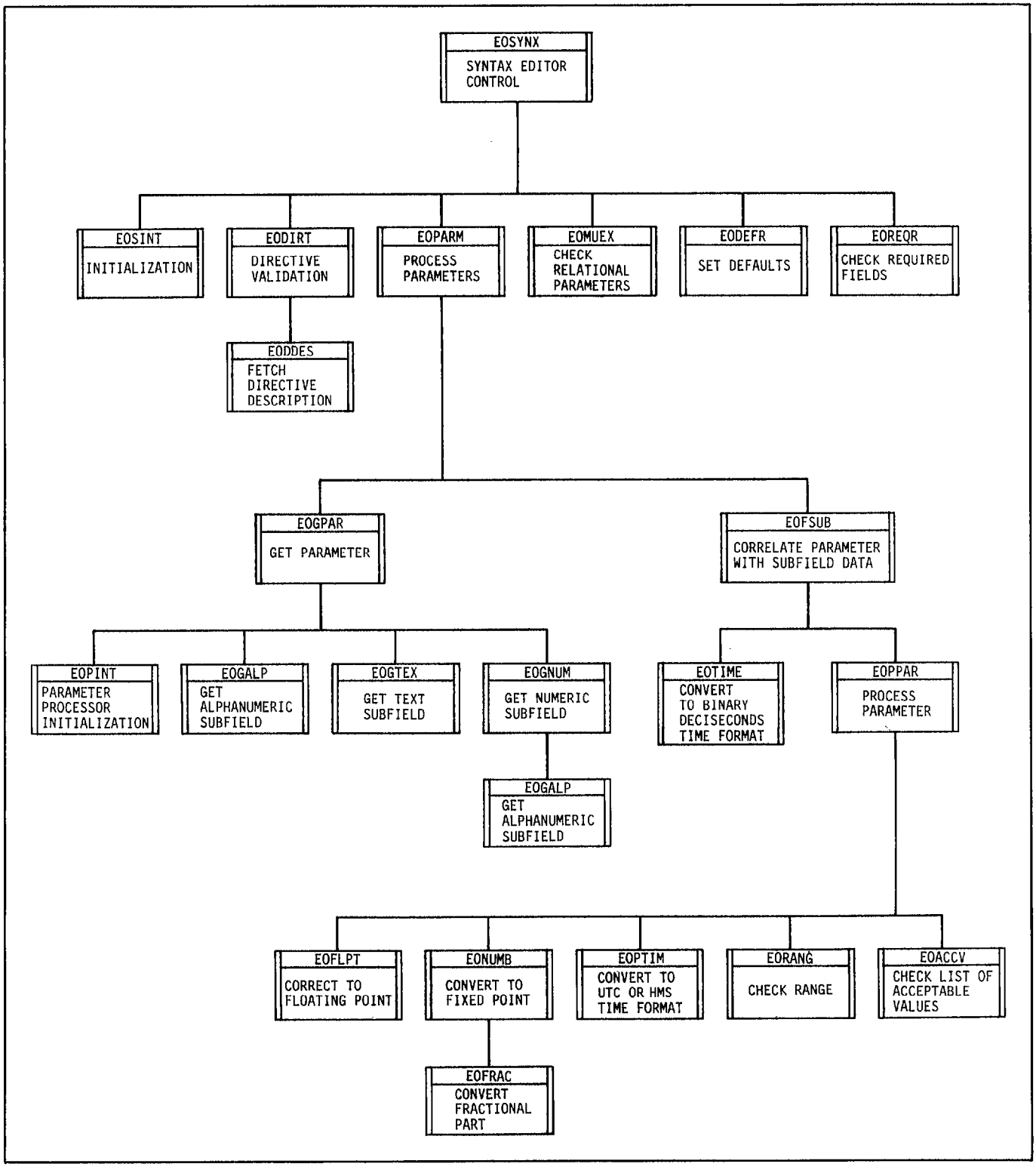

Fig. 8. Syntax editor hierarchical tree

IDP DESIGN CHECKLIST

1) Does IDP have to determine the particular case by checking to see which parameters or what values have been entered?

2) Should IDP perform directive error checks in addition to those that have been checked for by the Syntax Editor?

3) Is there any special processing (e.g., conversions, calculations) that the IDP must perform?

4) Should the IDP set flags or data in global or private common for the corresponding subsystem or task?

5) Should a semaphore (interlock) be checked and set before the data is setup?

6) Should IDP notify the subsystem or task of a new data entry or flag?

7) Should the IDP wait for a completion response from the subsystem or task?

8) Should the IDP check for abnormal returns from the subsystem or task (and then take appropriate action)?

9) Should the IDP provide default values for any of the subfields?

10) Is the IDP consistent with the interface as defined in the Syntax Output Buffer and Syntax Definition Forms?

11) Should the IDP issue an acknowledgement for completion of processing the directive?

12) Have the appropriate Syntax Output Buffer equivalences been defined and coded?

13) Are there any other functions the IDP should perform before exiting?

14) For assembly language IDPs, did the IDP preserve R10 and return to the Directive Processor via the address contained in R10?

Fig. 9. IDP design checklist

DIRECTIVE STATUS REPORT

Directive	Programmer	Directive Description	Step	Status
EHL	M. BURNS	Enables or disables halting of procedure processing when a procedure error has occurred	100	****
ELM	R. BILLINGS	Modifies a command file prior to transmission to the CPA	C070	***
END	T. OWENS	Terminates a test sequence	050	****
ERA	S. MAK	Erases file directory	C120	-
ESD	M. MACAULAY	Supplies the support data file name which is to be edited	L040	**
ESH	J. STICHT	Edits the header of the requested support data file name	L040	**
EST	J. STREIT	Enables or disables reporting, tolerance tests, status tests, or error rate tests and sets appropriate parameters	T080	*

Status:

- \- Proposed
- * Submitted
- ** Approved
- *** Implemented
- **** Demonstrated

Fig. 10. Directive status report

CDRR - ENABLE/DISABLE UDT/DDT OR ALL FOR GCF DSN BLOCK(S)

DESCRIPTION: The directive CDRR enables or disables GCF DSN block(s). The request enters into or removes from the routing table either 'all' prime CPA routing codes or a specific UDT/DDT code.

FORMAT: CDRR z, (<UDT=x, DDT=y>,ALL)

FIELDS: Three required field entries can be input in any sequence when specifying a UDT/DDT code. For this case the request is as follows:

CDRR z, UDT=x, DDT=y

Two fields in any sequence are required when requesting all prime CPA routing codes. The request is as follows:

CDRR z, ALL

NOTE: If the 'ALL' request is enabled, then the specifications for unique UDT/DDT pairs can not be requested. The 'ALL' request must be disabled before specific UDT/DDT pairs can be enabled. The converse is also true.

CAUTION: The UDT/DDT pairs entered are not checked for correctness or validity.

It is possible to lose a block if an identical enable request is entered. This occurs because the prior request is removed from the routing table to prevent reception of identical blocks.

PARAMETER CHARACTERISTICS:

Parameter	Description	Input Type	Range Or Maximum Length	Input Status (Required, Optional, Mutually Exclusive, Acceptable Values, Default Values)
z	Enable or disable block(s)	Alpha	1 Char.	Acceptable values: E - for enable D - for disable
UDT	User Dependent Type Identifier			Keyword for x
x	UDT Code	Integer	0 -127	Specifies unique UDT code
DDT	Data Dependent Type Identifier			Keyword for y
y	DDT code	Integer	0 - 127	Specifies unique DDT code
ALL	Self Identifying variable			Acceptable value: ALL - indicates all prime CPA routing codes

Fig. 11. Sample SOM directive description

```
EXAMPLES:
          Acceptable requests:
              CDRR E, UDT=#47, DDT=#66
              CDRR E, UDT=0107, DDT=0146
              CDRR E, UDT=71, DDT=102
              CDRR E, ALL
              CDRR D, UDT=#47, DDT=#66
          Rejected requests:
              CDRR E              ( One or more fields are missing. )
              CDRR E, UDT=ALL     ( All is not a specific UDT code.)
              CDRR E, UDT=#7A     ( A DDT code has not been specified.)
```

CDRR Directive Response Messages

MESSAGE	REASON	OPERATOR RESPONSE
ALL INVALID, UDT/ DDT PAIR(S) ENABLED	The request to enable the prime CPA routing codes has been rejected because specific UDT/DDT routing codes were previously enabled.	Disable each UDT/ DDT previously entered, reenter the 'all' request
ERROR, CRUN NOT ENTERED	The CRUN directive must be entered prior to CDRR to identify the mission.	Enter the CRUN directive and reenter CDRR
REQUEST NOT ACCOMPLISHED	A status check has indicated that the SPT EXEC could not execute the request.	Reenter the request
UDT/DDT INVALID, ALL ENABLED	The request to enable the specific UDT/DDT routing code has been rejected because the prime CPA routing codes are enabled.	Disable the prime CPA routing codes with the 'all' request, and reenter the directive for the specific UDT/DDT

Fig. 11 (contd)